ReAgent: Point Cloud Registration using Imitation and Reinforcement Learning

Dominik Bauer, Timothy Patten and Markus Vincze
TU Wien
Vienna, Austria
{bauer,patten,vincze}@acin.tuwien.ac.at

Abstract

Point cloud registration is a common step in many 3D computer vision tasks such as object pose estimation, where a 3D model is aligned to an observation. Classical registration methods generalize well to novel domains but fail when given a noisy observation or a bad initialization. Learning-based methods, in contrast, are more robust but lack in generalization capacity. We propose to consider iterative point cloud registration as a reinforcement learning task and, to this end, present a novel registration agent (ReAgent). We employ imitation learning to initialize its discrete registration policy based on a steady expert policy. Integration with policy optimization, based on our proposed alignment reward, further improves the agent's registration performance. We compare our approach to classical and learning-based registration methods on both ModelNet40 (synthetic) and ScanObjectNN (real data) and show that our ReAgent achieves state-of-the-art accuracy. The lightweight architecture of the agent, moreover, enables reduced inference time as compared to related approaches. ***Code is available at github.com/dornik/reagent.***

1. Introduction

Depending on the application domain, point cloud registration methods need to fulfill a range of properties. For example, AR applications and robotics applications require real-time inference speed and robustness to unexpected observations. In such real-world deployment, generalization to categories that were not seen during training is required. Further, registration approaches also need to generalize to different tasks, such as object pose estimation or scan alignment. Finally, an interaction with or scrutiny by a human might be required. For this, the method's steps need to be interpretable. These properties are often competing and thus difficult to achieve using a single approach.

As diverse as the required properties are the approaches that are proposed to solve point cloud registration. Distinctive features of proposed methods are global [28, 30] or local optimality [3], as well as one-shot [24] or iterative computation [1]. Global considerations allow for greater robustness to initial conditions than local methods, albeit at the cost of significantly increased computation time. While iterative methods may achieve higher accuracy than one-shot approaches through repeated registration, they may diverge over multiple steps. Furthermore, learning-based approaches are proposed in related work [24, 1, 25, 5], which are shown to be more robust to initialization and noise than classical approaches. However, these methods are not robust to domain change, e.g., when transferred to novel tasks.

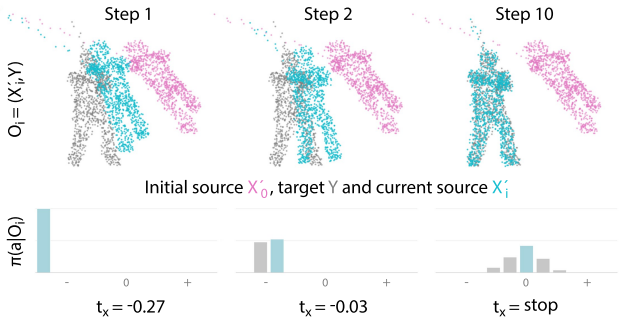

Figure 1: Iterative registration using ReAgent. The source point cloud (cyan) is aligned to the target point cloud (gray), starting from an initial source (magenta). ReAgent follows policy π by taking action $a_i = \arg\max_a \pi(a|O_i)$ given the current observation O_i, improving registration step-by-step.

In an effort to bridge this performance gap between methods, we design a novel registration approach that unifies accuracy, robustness to noise and initialization with inference speed. While reinforcement learning methods for RGB-based object pose refinement are proposed [22, 4], to the best of our knowledge, we are the first to consider 3D point cloud registration as a reinforcement learning problem. Our approach is based on a combination of Imitation Learning (IL) and Reinforcement Learning (RL); imitating an expert to learn an accurate initial policy, reinforcing a symmetry-invariant reward to further improve the policy.

As illustrated in Figure 1, the proposed registration agent (*ReAgent*) treats registration as an iterative classification of the observed point cloud pair into discrete steps. In summary, we

- combine imitation and reinforcement learning for accurate and robust point cloud registration,
- propose a lightweight agent for fast inference and interpretability using iterative discrete actions and
- improve accuracy compared to state-of-the-art methods on synthetic and real datasets, while reducing inference time compared to related approaches.

We discuss related point cloud registration and reinforcement learning methods in Section 2. Section 3 presents the proposed registration agent. Section 4 provides experiments on synthetic and real data. The impact of the presented contributions is discussed in Section 5. Section 6 concludes the paper. Additional details and results on the related object pose estimation task are found in the supplement.

2. Related Work

The presented approach is influenced by previous work in point cloud registration and work that applies reinforcement learning to the related task of object pose estimation.

Classical Point Cloud Registration: The most influential work in point cloud registration is Iterative Closest Point (ICP) [3]. First, the (closest) points in the source and target points clouds are matched. Then, a transformation that minimizes the error between the matched points is found in closed form. These steps are repeated until convergence. Many variants are proposed, e.g., considering surface normals [17], color [14] or using non-linear optimization [6].

ICP may only find local optima. A branch-and-bound variant [28] trades global optimality for increased runtime. Other global approaches use local features [18] with RANSAC or directly optimize a global objective [30]. TEASER [27] achieves high robustness to large amounts of outliers through a truncated least squares cost and allows to certify global optimality of the estimated registration.

Learning-based Point Cloud Registration: Recent approaches based on neural networks (NN) use the idea of ICP and its global variants. Local features are extracted to determine a matching between the input clouds. Using this matching, the transformation is found either in closed form using differentiable Weighted SVD [24, 25, 29] or by optimization using stochastic gradient descent [5]. This enables the definition of end-to-end learnable registration pipelines. Notably, the method by Yew and Lee [29] additionally uses surface normals to compute Point Pair Features (PPF) as input. While there is effort to extract more robust features [29, 5], these methods typically use secondary networks that predict the sharpness of the match matrix to deal with imperfect correspondences and outliers [25, 29].

In contrast, another class of NN-based methods uses global features that represent whole point clouds and as such are more robust to imperfect correspondences. Seminal work in this direction by Aoki et al. [1] poses iterative registration as the alignment of global features using an interpretation of the Lucas-Kanade algorithm. A deterministic formulation that replaces the approximate with an exact Jacobian is proposed in [12], which increases the stability of the approach. The method in [19] allows one-shot registration of global features. We show that global feature representations may be used as state representation in RL and learned jointly with the agent's policy.

Reinforcement Learning in Object Pose Estimation: In the related domain of object pose estimation, Krull et al. [10] use RL to find a policy that efficiently allocates refinement iterations to a pool of pose hypotheses. Closely related to our approach, in RGB-based object pose estimation [22, 4], RL is used to train policies that manipulate an object's pose. Based on 2D segmentation masks, these agents learn to predict discrete refinement actions. In contrast, we focus on learning registration actions from 3D point clouds – while RGB-based methods use pretrained optical flow estimation as state, we use siamese PointNets. Additionally, we integrate IL to quick-start training and stabilize RL.

3. Point Cloud Registration Agent

In the following, we present our novel point cloud registration approach, based on imitation and reinforcement learning, called *ReAgent*. For readers unfamiliar with point cloud registration, Section 3.1 gives a brief introduction. In Section 3.2, we propose the fast and interpretable network architecture of the agent. Sections 3.3 and 3.4 present the learning procedure that enables accurate and robust registration. In Section 3.5, we discuss design choices that facilitate generalization to novel test categories and real data.

3.1. Background: Point Cloud Registration

Assume two point clouds, the *source* X and *target* Y, that represent an object or scene surface are given. In the simplest case, both sets are identical but may in general only partially overlap. The observed source X' is offset by an unknown rigid transformation $T' = [R' \in SO(3), t' \in \mathbb{R}^3]$, where R' is a rotation matrix and t' a translation vector. We define the tuple $O = (X', Y)$ as the *observation*, where

$$X' = T' \otimes X. \tag{1}$$

Given O, the task of point cloud registration is to find a rigid transformation $\hat{T}$ such that

$$\hat{T} \otimes X' = X. \tag{2}$$

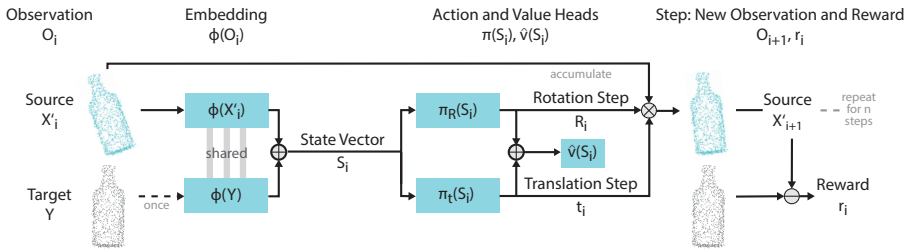

Figure 2: Architecture overview for one iteration of ReAgent.

The optimal transformation would thus be $\hat{T} = T'^{-1}$ such that we retrieve the original alignment (X, Y). In general, however, $\hat{T}$ is an error afflicted estimate of the registration. The target Y guides this registration process.

When n steps are taken to compute this transformation, this is referred to as *iterative registration*. In every step i, a rigid transformation $\hat{T}_i$ is estimated and the observed source is updated by

$$X'_i = \hat{T}_i \otimes X'_{i-1}. \tag{3}$$

The goal is that the final estimate after n steps is again

$$X'_n = \hat{T}_n \otimes ... \otimes \hat{T}_1 \otimes X'_0 = \hat{T} \otimes X' = X. \tag{4}$$

In a more general setting, one or both point clouds are noise afflicted. For example, their 3D coordinates may be jittered, their order may not correspond, not every point in the source may have a correspondence in the target and the number of points may not be identical. Such noise is typically due to the sensor recording the point clouds, varying view points or (self)occlusion.

3.2. The ReAgent Architecture

The proposed architecture for our point cloud registration agent is shown in Figure 2. The registration starts with a feature embedding that transfers the raw observed point clouds into global feature vectors. The concatenation of the source's and target's global feature vector is used as state representation, encoding the agent's information about the current registration state. A policy network uses the state representation to predict two action vectors, one for rotation and one for translation. Finally, the resulting transformation is applied to the observed source, iteratively improving the registration. In each such step, the agent receives a reward that judges how well it performs its task. The individual parts are now discussed in more detail.

Learned state representation: The observed source and target may have varying shape and may be noise afflicted. Our goal is to learn a more robust and more powerful representation than the bare point clouds. This is achieved by the feature embedding $\Phi(O)$, mapping from $N \times 3$ dimensional observation to $1 \times M$ dimensional state

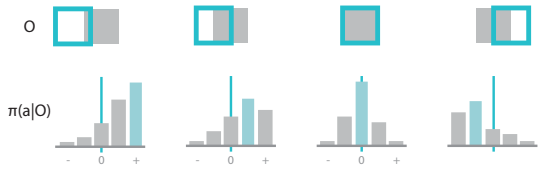

Figure 3: Illustration of interpretable actions. Top: Observed sources (cyan) with varying offset to the target (gray). Bottom: The probability of selecting each step size.

space. The source and target are passed through the embedding separately with shared weights. The concatenation of both global feature vectors is used as state S.

Discrete action space: We observe that, when trying to reach an exact registration in every iteration (i.e, by repeated one-shot registration), a bad estimate in one step may lead to divergence of the whole registration process. To this end, related work proposes to robustify the matching process [25, 29]. In an orthogonal approach, we aim to robustify the update steps themselves by using discrete, limited step sizes in each iteration. The discrete steps may be interpreted as the result of a classification of the observation into misalignment bins, as shown in Figure 3. Inspired by recent work by Shao et al. [22], we use a set of discrete steps along and about each axis as action space. We propose to use an exponential scale for the step sizes to quickly cover a large space during initial registration, while allowing accurate fine-registration in later steps.

Given state S, the agent's policy $\pi(S)$ gives the probability of selecting action a. The policy is computed by the agent's action head and predicts the step sizes for the iteration. Note that a is a vector of 6 sub-actions, one per rotation axis and translation axis. In addition, a value head estimates the baseline $\hat{v}(S)$. During the RL update, the baseline is subtracted from the returns of the actions to compute the advantage. This is commonly used to reduce variance as compared to using the returns directly.

Disentangled transformation: The concatenation of multiple rigid transformations with the source in Equation (4) may follow different conventions. The basic approach is to compute the matrix product of all $\hat{T}_i$ in homog-

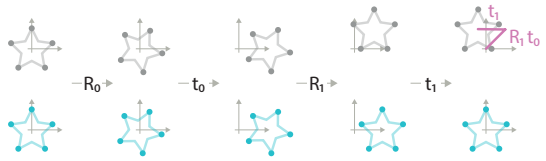

Figure 4: A transformation sequence and its effect on a point cloud using global (top) and disentangled transformation (bottom).

enized form and apply this to X'. Yet, as shown in Figure 4, when the rotation center is not the origin, a rotation induces an additional translation of the point cloud since

$$\begin{bmatrix} R_1 & t_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_0 & t_0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_1 R_0 & R_1 t_0 + t_1 \\ 0 & 1 \end{bmatrix}. \quad (5)$$

Note that this is equal to iterative application of $\hat{T}_i$ to X' as

$$R_1(R_0 X + t_0) + t_1 = R_1 R_0 X + R_1 t_0 + t_1. \quad (6)$$

To support interpretability, however, we would like a rotation action to only result in a local rotation of the object. Moreover, we want the rotation and translation axes to align with the global coordinate axes such that an action in a specific axis always results in the same global displacement. Formally, we want iterative transformations to result in

$$X_i = (\prod{}^i R_i) X + \sum{}^i t_i. \quad (7)$$

Such disentanglement of rotation and translation not only benefits interpretability but, as shown for image-based object pose estimation by Li et al. [13], is also beneficial for training of the agent; it does not need to account for the rotation-induced translation.

Following this idea, we propose a disentangled application of $\hat{T}$ to 3D point clouds. For iterative registration, we define the update rule for an accumulator $T_i = [R_i, t_i]$ by

$$R_i = \hat{R}_i R_{i-1}, \quad t_i = \hat{t}_i + t_{i-1}, \quad (8)$$

which is initialized with $T_0 = [I_{3\times3}, 0]$ and applied to the observed source by

$$X'_i = R_i(X' - \mu_{X'}) + \mu_{X'} + t_i. \quad (9)$$

Thereby, rotations are applied with the centroid of the observed source $\mu_{X'}$ as the origin. Since we only apply rigid transformations, the relation between points and the centroid does not change. The outcome is that no additional translation is introduced. This also holds when applying the accumulated transformation.

3.3. Imitating an Expert Policy

Learning a complex task, such as point cloud registration, from scratch using RL may take long to converge or may get stuck with a suboptimal policy; even more so if the state representation is learned jointly with the policy. To circumvent this issue, we initialize the state representation and the policy using IL.

In IL, the goal is to imitate the behavior of some domain expert. The simplest form of IL is Behavioral Cloning (BC). This assumes that, in every step, the agent has access to feedback from the expert. The feedback is used similarly to training data labels in supervised learning.

Expert policy: The expert feedback may come from interactions of a human expert or another algorithmic approach to solve the task. Since we can create training samples from point clouds by generating the initial rigid transformation, we have access to T'. We exploit this by defining two expert policies that reduce the true transformation error in the current step, given by

$$\delta_i^R = R' R_i^\top, \quad \delta_i^t = t'_d - t_i, \quad (10)$$

where t'_d is the disentangled form of t' that accounts for the translation induced by R' using

$$t'_d = t' - \mu_{X'} + R' \mu_{X'}. \quad (11)$$

The expert policy either takes the largest possible step that reduces the *absolute* error (greedy) or the *signed* error (steady). The steady expert produces trajectories with monotonously decreasing error, while the greedy expert produces optimal trajectories at the cost of oscillation.

Data gathering: The initial transformation alone is, however, insufficient to train our agent. The agent will observe certain trajectories during inference that are not covered by the generated initial errors T'. To this end, we rollout trajectories by following the stochastic policy of the agent to gather a replay buffer. The distribution of training data in this buffer is more representative of what the agent will observe during inference and dynamically adapts as the agent improves. By using the stochastic policy, we also guarantee exploration. As the training converges, the entropy of the policy $H(\pi)$ – and consequentially the exploration – reduces.

Behavioral cloning: The gathered training data, together with the annotation from the expert policy, allows us to train the agent using a 6-dimensional cross-entropy loss. For every observation, we gather registration trajectories. Once a certain number of trajectories is reached, the agent is updated using mini-batches from the shuffled buffer.

3.4. Improving through Reinforcement

The resulting agent policy is in two ways limited by the expert. On one hand, the agent cannot find a better policy than the expert as its actions would differ from the expert labels. On the other hand, different transformations will, in general, lead to different expert actions. If the observed

sources are, however, indistinguishable due to symmetry, they might be represented by an identical state vector. This hinders training of the agent as this would require different actions to follow from the same state vector.

Reward function: The overall goal of the agent is to align source and target. Following the expert policy, this alignment will reflect the initial transformation T'. Rather, the alignment should be treated equally for equivalent transformations $\tilde{T}'$ that result in indistinguishable observations where $X' \sim \tilde{X}'$. Instead of training the agent to exactly imitate the expert policy, we thus additionally use RL and define the training objective by a reward function.

In the proposed RL task, equal consideration of equivalent transformations is achieved by using the mean Chamfer distance (CD) between the currently observed source X' and true source $X = T'^{-1}X'$. This measure is insensitive to transformations that result in the same distance between closest points, e.g., rotations about a symmetry axis. Note, though, that the sampling rate of the point cloud may introduce fluctuations as X' moves through undersampled regions of X and vice-versa. However, this effect is lessened by considering the mean over all distances. Based on this, we define a step-wise reward

$$r = \begin{cases} -\varepsilon^-, & CD(X'_i, X) > CD(X'_{i-1}, X) \\ -\varepsilon^0, & CD(X'_i, X) = CD(X'_{i-1}, X) \\ \varepsilon^+, & CD(X'_i, X) < CD(X'_{i-1}, X). \end{cases} \quad (12)$$

Steps that reduce CD are rewarded by ε^+, "stop" gets a negative penalty $-\varepsilon^0$ to discourage pausing and divergent steps are penalized by $-\varepsilon^-$. We choose $\varepsilon^- > \varepsilon^+$ to discourage alternating diverging and converging steps.

Policy optimization: The policy learned using IL already performs accurate registration. Large changes to the policy due to RL might result in forgetting and thereby worsening of the agent's performance. Rather, we want the policy after an RL update to be close to the previous policy. This is achieved by trust-region approaches such as Proximal Policy Optimization (PPO) [21]. The main idea of the clipped version of PPO is to limit the ratio between the previous and the updated policy by a fixed threshold. In addition, as observed in related work combining BC and GAIL [8], it is benefitial to jointly optimize BC and RL objectives as to further limit divergence of the policy. In our combined approach, both IL and RL use the same replay buffer. Since the RL term considers equivalent transformations, the agent is able to differentiate between bad steps (discouraged by IL and RL), equivalent steps (discouraged by IL, encouraged by RL) and the best steps (encouraged by IL and RL).

3.5. Implementation Details

The final combination of IL and RL that is used to train the agent is presented in Algorithm 1, where *agent* implements the architecture shown in Figure 2.

Algorithm 1 Combined Imitation and Reinforcement Learning using a Replay Buffer

for all observations O in $\mathcal{O}$ **do**
  % Gather replay buffer
  for N trajectories **do**
    for n refinement steps **do**
      agent predicts policy $\pi(O)$ and value $\hat{v}$
      action a is sampled from policy $\pi(O)$
      take action a, receive reward r and next O'
      add sample to buffer b, step observation $O = O'$
    end for
  end for
  % Process replay buffer
  compute return R, shuffle buffer b
  for all samples in buffer b **do**
    agent predicts new policy $\pi'(O)$ and value $\hat{v}'$
    % Imitate expert
    expert predicts action a^*
    compute cross-entropy loss l_{IL} of $\pi'(O)$ and a^*
    % Reinforce
    compute PPO loss l_{RL} of $\pi'(O)$ and $\pi(O)$
    % Update agent
    $l = l_{IL} + l_{RL} \cdot \alpha$
    backpropagate combined loss l
  end for
  clear buffer b
end for

Agent: We choose a PointNet-like architecture [15] as feature embedding. As indicated by the findings of Aoki et al. [1], the T-nets in the original PointNet architecture are unnecessary for registration and are therefore omitted. We further observe that a reduced number of embedding layers is sufficient to learn an expressive embedding. The feature embedding Φ therefore reduces to 1D convolution layers of size $[64, 128, 1024]$, followed by max pooling as symmetric function. The concatenation of these 1024 dimensional global features gives a 2048 dimensional state vector.

In each iteration, the policy gives a step size for all 6 degrees of freedom. This is implemented as a prediction of the logits of a multi-categorical distribution. There is a total of 11 step sizes per axis: $[0.0033, 0.01, 0.03, 0.09, 0.27]$ in positive and negative direction, as well as a "stop" step. For rotation, step sizes are interpreted in radians.

Shao et al. [22] propose to use shared initial layers for the action and value heads in an actor-critic design. We adapt this approach to our architecture and implement each head as fully-connected layers of size $[512, 256, D]$, where D is 33 for rotation and translation estimation and 1 for the value estimate. The concatenation of the middle layer of both action heads serves as input to the value head.

Expert: While the greedy policy achieves a lower error, when used to train the agent, both experts result in the same agent accuracy. We thus favor the more interpretable trajectories learned from the steady policy.

PPO: We use the PPO formulation from [21] for actor-

critic architectures with an entropy term that encourages exploration. The advantage $\hat{A}$ in the PPO loss uses the agent's value estimate and Generalized Advantage Estimation (GAE) [20]. For the reward function, we experimentally determine $(\varepsilon^+, \varepsilon^0, \varepsilon^-) = (0.5, 0.1, 0.6)$ to successfully guide the agent.

Hyperparameters: Further parameters are the number of registration steps $n = 10$, the number of trajectories per update $N = 4$, the discount factor $\gamma = 0.99$ and the GAE factor $\lambda = 0.95$. The RL loss term is scaled by $\alpha = 2$.

Regularization: While related methods use weight decay, batch or layer normalization for regularization [1, 24], we observe that affine data augmentation achieved better results with our architecture. Namely, we use 1) random scaling sampled from $\mathcal{N}(1, 0.1)$, clipped to $[0.5, 1.5]$, 2) shearing in uniformly random direction by a random angle sampled from $\mathcal{N}(0, 5)$, clipped to $[-15, 15]\,\mathrm{deg}$ and 3) mirroring about a plane with uniformly random normal.

4. Experiments

In the following, we evaluate the proposed point cloud registration agent on synthetic and real data. To evaluate our initial design goals of accuracy, inference speed and robustness, we consider noise-afflicted conditions on synthetic data. The generality of the approach is shown by results on held-out categories on synthetic data and the transfer to real data.

Baselines: For comparison, we evaluate two classical and two learning-based approaches. The former are Point-to-Point Iterative Closest Point (ICP) [3] and Fast Global Registration (FGR) [30], both as implemented in Open3D [31]. PointNetLK [1] is an iterative approach based on global PointNet features. As with our approach, we set the number of iterations to 10. Deep Closest Point with Transformer (DCP-v2) [24] is a local feature-based approach, predicting one-shot registration. As the latter methods provide no pretrained models on the ModelNet40 category splits, we retrain them using the published code.

Datasets: As in prior work, we evaluate on ModelNet40 [26], which features synthetic point clouds sampled from CAD models. To additionally evaluate performance on real data, we provide results on ScanObjectNN [23], featuring observations captured from an RGB-D sensor. Note that all learning-based methods (including ours) use only 3D coordinates, while the FPFH features used by FGR additionally require surface normals. On ModelNet40, the models' normals are used; on ScanObjectNN, the normals are computed from the respective observations.

Metrics: In line with prior work [24, 25], we provide the Mean Average Error (MAE) over Euler angles and translations. Yew and Lee [29] propose to additionally evaluate the isotropic error for rotation and translation (ISO), as well as a modified Chamfer distance ($\tilde{CD}$) to cover symmetric ambiguity. The isotropic rotation error is computed by the geodesic distance between the rotation matrices and the isotropic translation error uses the Euclidean norm. All angles are given in degrees. Moreover, we provide the area under the precision-recall curve (AUC) for the Average Distance of Model Points with Indistinguishable Views (ADI) [7], a metric commonly used in object pose estimation. The ADI is normalized to the model diameter and we clip at a precision threshold of 10% of the diameter. Note that $\tilde{CD}$ and ADI AUC implicitly consider symmetry.

Training: All methods are evaluated using an Intel Core i7-7700K and an NVIDIA GTX 1080. We train the proposed agent using Adam [9] with AMSGrad [16] and a batch size of 32. The replay buffer contains 4 trajectories of 10 steps each, resulting in a total of 1280 observations. We pretrain the agent for 50 epochs using IL ($\alpha = 0$) on clean point clouds from ModelNet40. During pretraining, we start with a learning rate of $1e^{-3}$, and halve it every 10 epochs. We then fine-tune the policy for an additional 50 epochs on the first 20 categories of ModelNet40 with the noise defined in Section 4.1. Fine-tuning uses the same learning rate schedule, albeit starting from $1e^{-4}$. We provide separate results for training with only IL (*ours IL*) and using the combined approach (*ours IL+RL*). Note that this policy is used for all experiments and thus shows the generalization performance of the proposed approach.

4.1. Synthetic Data: ModelNet40

To validate generalization to unseen points clouds and novel categories, we follow related work [1, 24, 25] and use the point clouds generated by [15] based on ModelNet40. All approaches are trained on the training split of the first 20 categories. The data augmentations follow related work [29]: Of the 2048 points, 1024 are randomly and independently subsampled for source and target to introduce imperfect correspondences. The source is transformed by a random rotation R' of $[0, 45]\,\mathrm{deg}$ per-axis and a random translation t' of $[-0.5, 0.5]$ per-axis. Random noise is sampled (again independently for source and target) from $\mathcal{N}(0, 0.01)$, clipped to 0.05 and applied to the point clouds. Finally, the point clouds are shuffled as to permute the order of points. Table 1 (left) shows results on the test split of the first 20 categories. Table 1 (right) shows results on the test split of the second 20 categories. For consistency, we train the other learning-based approaches in the noisy condition.

As shown in Table 1, our approach successfully generalizes to novel point clouds and novel categories. We report improved accuracy across all metrics as compared to related work. The comparison in the rightmost column shows that our approach is also the fastest of the evaluated learning-based point cloud registration methods. Inference speed is even comparable to the one-shot method DCP-v2. However, the performance of DCP-v2 deteriorates with imper-

	Held-out Models						Held-out Categories						
	MAE (↓)		ISO (↓)		ADI (↑)	$\tilde{CD}$ (↓)	MAE (↓)		ISO (↓)		ADI (↑)	$\tilde{CD}$ (↓)	T (↓)
	R	t	R	t	AUC	$\times 1e^{-3}$	R	t	R	t	AUC	$\times 1e^{-3}$	[ms]
ICP	3.59	0.028	7.81	0.063	90.6	3.49	3.41	0.024	7.00	0.051	90.5	3.08	**9**
FGR$^+$	2.52	0.016	4.37	0.034	92.1	1.59	1.68	0.011	2.94	0.024	92.7	1.24	68
DCP-v2	3.48	0.025	7.01	0.052	85.8	2.52	4.51	0.031	8.89	0.064	82.3	3.74	23
PointNetLK	1.64	0.012	3.33	0.026	93.0	1.03	1.61	0.013	3.22	0.028	91.6	1.51	45
ours IL	**1.46**	**0.011**	**2.82**	**0.023**	**94.5**	**0.75**	1.38	0.010	2.59	**0.020**	**93.5**	**0.95**	21
ours IL+RL	1.47	**0.011**	2.87	**0.023**	94.5	**0.75**	**1.34**	**0.009**	**2.48**	**0.020**	93.3	0.99	

Table 1: Results on ModelNet40 with held-out point clouds from categories 1-20 (left) and on held-out categories 21-40 (right). Note that ↓ indicates that smaller values are better. Runtimes are for a single registration with 1024 points per cloud. $^+$ indicates that FGR additionally uses normals, while the remaining methods only use 3D coordinates.

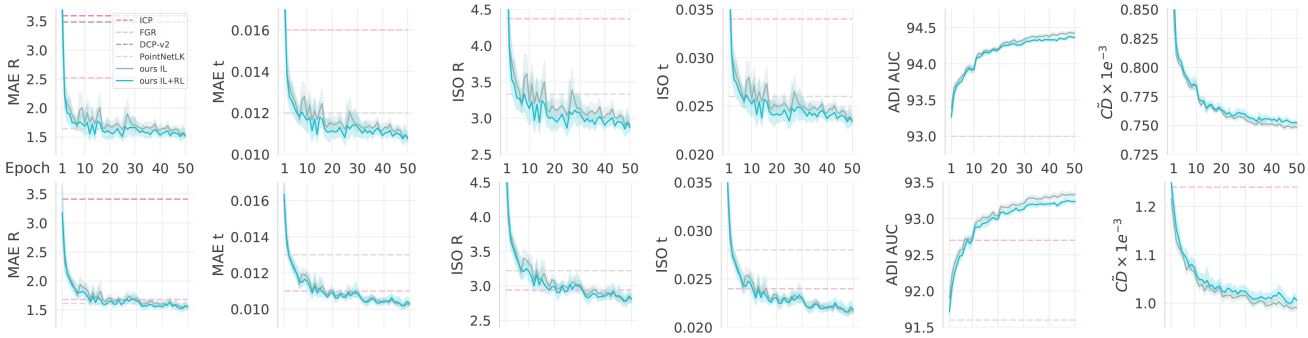

Figure 5: Convergence of ReAgent with 10 random seeds on held-out models (top) and categories (bottom) of ModelNet40. The lines show the mean and the shaded areas indicate the 95%-confidence intervals. Best viewed digitally.

	Segmented Objects						
	MAE (↓)		ISO (↓)		ADI (↑)	$\tilde{CD}$ (↓)	T (↓)
	R	t	R	t	AUC	$\times 1e^{-3}$	[ms]
ICP	5.34	0.036	10.47	0.076	88.1	2.99	**19**
FGR$^+$	**0.11**	**0.001**	**0.19**	**0.001**	**99.7**	**0.16**	131
DCP-v2	7.42	0.050	14.93	0.102	72.4	4.93	54
PointNetLK	0.90	0.010	1.74	0.020	92.5	1.09	45
ours IL	0.77	0.006	1.33	0.012	95.7	0.30	21
ours IL+RL	0.93	0.007	1.66	0.014	95.4	0.34	

Table 2: Results on ScanObjectNN with the object segmented from the observation. Learning-based methods use the model trained on ModelNet40. Note that ↓ indicates that smaller values are better. Runtimes are for a single registration and 2048 points per cloud. $^+$ indicates that FGR additionally uses normals.

fect correspondences, as is the case with noisy observations.

When generalizing to held-out models, as shown in Figure 5, the addition of RL successfully improves accuracy on the rotation-based metrics. As indistinguishable observations are due to rotations in this scenario these benefit most from the policy optimization. Yet, this improvement diminishes with novel categories, also indicated by the results on ScanObjectNN. Surprisingly, the consideration of symmetry via RL even slightly decreases mean performance on ADI AUC and $\tilde{CD}$ over 10 random seeds.

4.2. Real Data: ScanObjectNN

To evaluate generalization from synthetic to real data, we use the point clouds with segmented objects from ScanObjectNN dataset with 2048 points each. The same type of rigid transformations as in the previous condition are applied to the source. No additional noise is applied as the dataset already represents the characteristics of a specific depth sensor. For learning-based methods, the same models as in the previous conditions are used without any retraining or fine-tuning.

As shown in Table 2, our approach transfers from training on ModelNet40 to testing on ScanObjectNN with high accuracy. Only FGR, additionally using normals to compute FPFH features, performs consistently better under this condition. However, inference time of FGR is almost 6 times higher compared to our approach. Notably, the inference time of PointNetLK and of our approach is barely affected by the doubling of the number of points. While DCP-v2 requires repeated neighborhood computation that negatively affects inference time, both PointNet-based approaches benefit from the independent embedding per point. Qualitative examples are shown in Figure 6.

Further experiments on real data and the related object pose estimation task are found in the supplement.

Data		Expert		Φ, π		T		$\tilde{CD}$
stoch.	augm.	greedy	steady	deep	wide	basic	disent.	$\times 1e^{-4}$ (↓)
	✓		✓		✓		global	42.08
✓			✓		✓		global	11.81
✓	✓	✓			✓		global	2.61
✓	✓		✓	✓			global	2.92
✓	✓		✓		✓	✓		3.90
✓	✓		✓		✓		local	3.74
✓	✓		✓		✓		global	2.63

Table 3: Ablation study. Results of *ours IL*, pretrained on clean point clouds from held-out ModelNet40 categories.

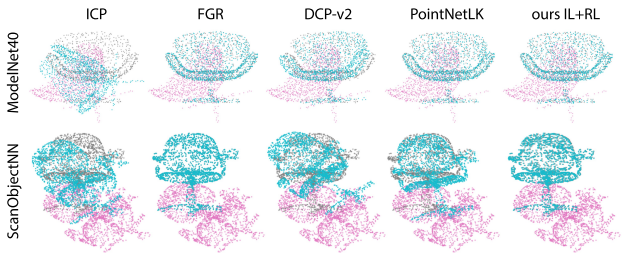

Figure 6: Qualitative examples. Columns show target (gray), initial (magenta) and registered source (cyan).

5. Discussion

In this section, we motivate several key design choices. We discuss current limitations of the proposed approach and suggest avenues for future work.

5.1. Design of the Agent

Table 3 presents results for central design choices. As shown, the use of the stochastic agent policy to gather additional training data (stoch.) is essential to the success of the method. To a lesser extent, the regularization by augmenting data through affine transformations (augm.) prevents overfitting. In Section 3.3, the steady policy was already suggested to be more interpretable than the greedy policy. Even though the greedy policy on its own is more accurate than the steady policy, the results in Table 3 show that the agent trained using a steady policy achieves equally high accuracy. We support our choice to reduce the depth of the embedding (deep) and, instead, increasing the width of the head networks (wide) by the slightly increased accuracy. Finally, Table 3 highlights the importance of the representation and application of transformations. There is only a slight improvement by using a disentangled representation with rotations applied locally ($R_i = R_{i-1}\hat{R}_i$) as compared to using homogenized transformation matrices (basic). Using globally applied disentangled rotations ($R_i = \hat{R}_i R_{i-1}$), as suggested in Section 3.2, improves both accuracy and interpretability of our agent. By using a disentangled representation, the agent does not need to account for the rotation-induced translation. With global rotations, additionally, the rotation axes remain aligned with the global coordinate axes throughout trajectories.

5.2. Limitations and Future Work

Discrete steps induce finite accuracy, bound by the smallest step size. Similarly, the largest step size bounds the initial error that may be overcome by the agent in a given number of iterations. To further generalize our approach, a dynamically predicted scale factor could adapt step sizes.

For application to object pose estimation, a semantic segmentation head as proposed in the original PointNet paper [15] may be adapted to iteratively improve object segmentation during refinement. In addition, the combination of ReAgent with rendering-based verification such as proposed in [2] should be explored to efficiently consider multiple initial pose estimates.

Further applications of the proposed method, such as self localization in large maps, will require efficient means to determine a state embedding from vast numbers of points. Replacing the PointNet embedding with (Deep) Lean Point Networks [11] would increase model capacity, allowing transfer to such complex domains.

6. Conclusion

We present a novel point cloud registration agent, called ReAgent, that leverages imitation and reinforcement learning to achieve accurate and robust registration of 3D point clouds. Its discrete actions and steady registration trajectories are shown to be interpretable, while achieving fast inference times. The generality of our approach is evaluated in experiments on synthetic and real data. On the synthetic ModelNet40 dataset, our approach outperforms all evaluated classical and learning-based state-of-the-art registration methods. On ScanObjectNN, featuring real data, our approach achieves state-of-the-art for all comparable methods that only use 3D coordinates. While introducing normals, such as in FGR, achieves slightly better registration accuracy, our method is 6 times faster, making it more suitable for real-time applications. We believe that these properties make *ReAgent* useful in many application domains, such as object pose estimation or scan alignment.

Acknowledgements: This work was supported by the TU Wien Doctoral College TrustRobots and the Austrian Science Fund (FWF) under grant agreements No. I3968-N30 HEAP and No. I3969-N30 InDex. We would like to thank Matthias Hirschmanner, Kiru Park and Jean-Baptiste Weibel for valuable feedback and insightful discussions.

References

[1] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. Pointnetlk: Robust & efficient point cloud registration using pointnet. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7163–7172, 2019. 1, 2, 5, 6

[2] Dominik Bauer, Timothy Patten, and Markus Vincze. Verefine: Integrating object pose verification with physics-guided iterative refinement. *IEEE Robot. Autom. Lett.*, 5(3):4289–4296, 2020. 8

[3] Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, 1992. 1, 2, 6

[4] Benjamin Busam, Hyun Jun Jung, and Nassir Navab. I like to move it: 6d pose estimation as an action decision process. *arXiv preprint arXiv:2009.12678*, 2020. 1, 2

[5] Christopher Choy, Wei Dong, and Vladlen Koltun. Deep global registration. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2514–2523, 2020. 1, 2

[6] Andrew W Fitzgibbon. Robust registration of 2d and 3d point sets. *Image and Vision Computing*, 21(13-14):1145–1153, 2003. 2

[7] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *ACCV*, pages 548–562, 2012. 6

[8] Rohit Jena and Katia Sycara. Loss-annealed gail for sample efficient and stable imitation learning. *arXiv preprint arXiv:2001.07798*, 2020. 5

[9] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[10] Alexander Krull, Eric Brachmann, Sebastian Nowozin, Frank Michel, Jamie Shotton, and Carsten Rother. Poseagent: Budget-constrained 6d object pose estimation via reinforcement learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6702–6710, 2017. 2

[11] Eric-Tuan Le, Iasonas Kokkinos, and Niloy J Mitra. Going deeper with lean point networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 9503–9512, 2020. 8

[12] Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Deterministic pointnetlk for generalized registration. *arXiv preprint arXiv:2008.09527*, 2020. 2

[13] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6d pose estimation. In *Eur. Conf. Comput. Vis.*, pages 683–698, 2018. 4

[14] Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Colored point cloud registration revisited. In *Int. Conf. Comput. Vis.*, pages 143–152, 2017. 2

[15] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 652–660, 2017. 5, 6, 8

[16] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019. 6

[17] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *IEEE Int. Conf. 3-D Digital Imaging and Modeling*, pages 145–152, 2001. 2

[18] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Int. Conf. Robot. Autom.*, pages 3212–3217. IEEE, 2009. 2

[19] Vinit Sarode, Xueqian Li, Hunter Goforth, Yasuhiro Aoki, Animesh Dhagat, Rangaprasad Arun Srivatsan, Simon Lucey, and Howie Choset. One framework to register them all: Pointnet encoding for point cloud alignment. *arXiv preprint arXiv:1912.05766*, 2019. 2

[20] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015. 6

[21] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 5

[22] Jianzhun Shao, Yuhang Jiang, Gu Wang, Zhigang Li, and Xiangyang Ji. Pfrl: Pose-free reinforcement learning for 6d pose estimation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 11454–11463, 2020. 1, 2, 3, 5

[23] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Int. Conf. Comput. Vis.*, pages 1588–1597, 2019. 6

[24] Yue Wang and Justin M Solomon. Deep closest point: Learning representations for point cloud registration. In *Int. Conf. Comput. Vis.*, pages 3523–3532, 2019. 1, 2, 6

[25] Yue Wang and Justin M Solomon. Prnet: Self-supervised learning for partial-to-partial registration. In *Adv. Neural Inform. Process. Syst.*, pages 8814–8826, 2019. 1, 2, 3, 6

[26] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1912–1920, 2015. 6

[27] Heng Yang, Jingnan Shi, and Luca Carlone. Teaser: Fast and certifiable point cloud registration. *IEEE Trans. Robot.*, 2020. 2

[28] Jiaolong Yang, Hongdong Li, Dylan Campbell, and Yunde Jia. Go-icp: A globally optimal solution to 3d icp point-set registration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(11):2241–2254, 2015. 1, 2

[29] Zi Jian Yew and Gim Hee Lee. Rpm-net: Robust point matching using learned features. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 11824–11833, 2020. 2, 3, 6

[30] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *Eur. Conf. Comput. Vis.*, pages 766–782. Springer, 2016. 1, 2, 6

[31] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018. 6